

A New Spectral Method for Latent Variable Models

Matteo Ruffini, Marta Casanellas, and Ricard Gavalda

Universitat Politècnica de Catalunya

December 13, 2016

Abstract

This paper presents an algorithm for the unsupervised learning of latent variable models from unlabeled sets of data. We base our technique on spectral decomposition, providing a technique that proves to be robust both in theory and in practice. We also describe how to use this algorithm to learn the parameters of two well known text mining models: single topic model and Latent Dirichlet Allocation, providing in both cases an efficient technique to retrieve the parameters to feed the algorithm. We compare the results of our algorithm with those of existing algorithms on synthetic data, and we provide examples of applications to real world text corpora for both single topic model and LDA, obtaining meaningful results.

1 Introduction

Latent variable models (LVM) are a wide class of parametric models characterized by the presence of some hidden *unobservable* variables influencing observable data. A lot of widely used models belong to this class: Gaussian Mixtures, Latent Dirichlet Allocation, Naïve Bayes and Hidden Markov Models and many others; in recent years, they have been object of an increasing interest in the learning literature due to the widespread of real world applications, from health-care data mining to text analytics. The huge availability of data, consequence of the development of new technologies, has boosted the need for efficient and fast algorithms to learn models belonging to this class.

Each LVM is designed as a set of observable variables (called *features*) and a set of hidden variables that influence the first. Learning a LVM means, given the model structure and a sample, to infer the parameters that characterize the hidden variables and their relation with the observable features.

The classical approach was the Expectation Maximization method (EM) (Dempster et al., 1977), which has been widely used because of its generality and easiness of implementation; however EM is known to produce suboptimal results and might be very slow when the model dimension grows (Balle et al., 2014). To overcome these issues a variety of methods exploiting tensor analysis and spectral decomposition have been recently proposed to learn various LVM: such as (Dasgupta, 1999; Sanjeev and Kannan, 2001; Dasgupta and Schulman, 2007; Vempala and Wang, 2002; Belkin and Sinha, 2010; Kalai et al., 2010; Moitra and Valiant, 2010; Hsu and Kakade, 2013) for mixture models or (Mossel and Roch, 2005; Hsu et al., 2012) for Hidden Markov Models. In Anandkumar et al.

E-mail: matteo.ruffini@estudiant.upc.edu, marta.casanellas@upc.edu, gavalda@cs.upc.edu

(2014), the authors presented an exhaustive survey showing that the spectral learning of most of the known LVM could be abstracted in two steps: first, given a prescribed LVM, they show how to operate with the low-order moments of the observable data in order to obtain a symmetric, low rank three dimensional tensor; as a second step this tensor is decomposed to obtain the unknown parameters of the model. That paper accurately describes how to transform moments to obtain a symmetric tensor representation for various LVM, and provides also one of the most popular methods to decompose the retrieved tensor and obtain the unknown model parameters: an iterative technique named *tensor power method*. Practically speaking, the tensor power method is very difficult to implement, requiring hundreds of lines of code even for a naive implementation. Also, as an iterative method, it requires a tuning of the convergence hyperparameters (the parameters that determine the number of iterations); to properly set these values is a non-trivial task, requiring a series of trials and errors; in our experience, it is not uncommon to get poor results from the algorithm when parameters are tuned poorly. Alternative decomposition methods are proposed in Anandkumar et al. (2012a), where an algorithm based on the eigenvectors of a linear operator is used, and in Anandkumar et al. (2012b) with a method based on the singular vectors of a singular values decomposition (SVD); both these techniques are randomized, as they rely on random matrices whose values have a non-negligible impact on the final results.

Moving to the applications side, LVM are very popular for text mining: here the observable data, called *features* of the model, is generally considered to be the words appearing in a document, while the hidden variable can be, for example, the topic of the document. A popular model for unsupervised topic mining is the single topic model, where each text is assumed to deal with a unique topic and the probability of a given word of belonging to a text depends on the topic itself of the text. An alternative and more complex method is Latent Dirichlet Allocation (LDA) (see Griffiths and Steyvers, 2004; Blei et al., 2003), where each text deals with more than one topic; words appear in the text according to the proportions of the topics present in the text. When using spectral methods, the standard procedure to learn these models consists in manipulating the observable moments of the data to obtain a set of symmetric, low rank tensors (examples are proposed in Anandkumar et al. 2014), and then retrieve the model parameters decomposing the retrieved tensors with a decomposition algorithm.

The contributions of this paper are the following:

- We formally present a technique to retrieve a low-rank symmetric tensor representation for the single topic model and Latent Dirichlet Allocation. This method was already sketched in Zou et al. (2013), without any study of accuracy of moment estimates and only for the single topic model. In this paper, we present an analysis of how the accuracy of the estimates depends on the sample and we extend the method to Latent Dirichlet Allocation.
- We provide a new algorithm (named SVTD, *Singular Value based Tensor Decomposition*) to decompose the retrieved low-rank symmetric tensor, alternative to the ones presented in the cited literature. Our algorithm is simple (it requires few lines of code to be implemented), deterministic (unlike those from Anandkumar et al. 2012b and Anandkumar et al. 2012a) and non-iterative (unlike the tensor power method in Anandkumar et al. 2014). It relies on the singular values of the SVD, which are known to be stable under random perturbations (unlike the singular vectors, as shown in Stewart 1998). In our opinion, the presented algorithm

might be a useful alternative to existing methods, as it seems very robust in the experiments, and can be easily implemented by practitioners. Furthermore, it does not include iterative techniques, so no hyperparameter tuning is needed. We also provide a perturbation analysis for this algorithm. In Remark 6 we outline in more detail the differences between the presented method and the state-of the art techniques.

- We then compare the performance of SVTD with those from the state of the art literature on synthetic data; we show that it performs at least as well as the existing methods, without having the drawbacks typical of randomized and iterative techniques. Furthermore, in our experiments, our method provides results that are more stable under perturbations, as a consequence of its deterministic nature. Finally, we test SVTD on real world text corpora, both for single topic model and LDA, with satisfactory and meaningful results. We remark that very few (none, in our knowledge) examples of text topic inference using LDA learned with spectral methods can be found in literature, so we believe to be the first in presenting a practical example of topic mixtures inference in an LDA model when parameters are learned with a spectral method (see Section 6.2.1).

The outline of the paper is the following: Section 2 and 3 contain the description of the proposed technique to retrieve a low-rank symmetric tensor representation for the single topic model; Section 4 contains the proposed decomposition algorithm; Section 5 contains a perturbation analysis; Section 6 tests the presented algorithm on both synthetic and real world data; Section 7 concludes the paper outlining possible future developments and applications.

In Sections 2 and 3 we describe a technique to retrieve a low-rank symmetric tensor representation for single topic model and for LDA; while the algorithm presented in this paper is general and can be used to learn many LVMs, it is useful to present these cases of application.

2 The Single Topic Model

We consider a corpus of N text documents and a set of k topics; each document is deemed to belong to only one topic. The vocabulary appearing in the corpus is constituted of n words, from which it is immediate to label all the words of the vocabulary with a number between 1 and n . The generative process works as follows:

- First, a (hidden) topic $Y \in \{1, \dots, k\}$ is drawn, according to a given probability distribution; we define, for any $j \in \{1, \dots, k\}$ the probability of drawing the topic j as follows:

$$\omega_j := \mathbb{P}(Y = j), \text{ and } \Omega := (\omega_1, \dots, \omega_k)'.$$

- Once the topic has been chosen, all the words of the documents are generated according to a multinomial distribution; for each $i \in \{1, \dots, n\}$, $\mu_{i,j}$ will be the probability of generating word i under topic j :

$$\mathbb{P}(\text{Drawing word } i | Y = j) = \mu_{i,j}, \text{ and } M = (\mu_{i,j})_{i,j} \in \mathbb{R}^{n \times k}.$$

Also we will denote with μ_i the set of columns of M :

$$M = [\mu_1 | \dots | \mu_k].$$

It is a common practice to identify a topic with the probability distribution of the words under that topic, i.e. with the columns μ_1, \dots, μ_k of M .

A practical encoding of the words in a document consists in identifying each word with an n -tuple $x \in \mathbb{R}^n$, defined as:

$$(x)_h := \begin{cases} 1 & \text{the word is } h, \\ 0 & \text{else.} \end{cases}$$

In fact, if x_j is the j -th word of a document of c words, we can define X as a vector whose coordinate h represents the number of times the word h has appeared in the document:

$$X := \sum_{j=1}^c x_j$$

It is common to call X a *bag of words* representation of a document. We can see that, if the topic is j , each coordinate of X is distributed as a binomial distribution with parameter c and $\mu_{i,j}$:

$$\text{Distr}(X_i|Y = j) \approx B(c, \mu_{i,j})$$

We now assume to have a corpus of N documents; for each document $i \in \{1, \dots, N\}$ we assume to have the word-count vector $X^{(i)}$, and the total number of words in the document:

$$c_i = \sum_{j=1}^n (X^{(i)})_j$$

These are the only variables that we assume known, while all the parameters of the model, i.e. the pair (M, Ω) , and the hidden topic of each document are supposed to be unknown.

Remark 1. Recovering the model parameters is a useful step to infer the hidden topic of each document in a corpus. In fact, given a set of parameters (M, Ω) , and a document X , if Y is the hidden topic of X we can calculate

$$\mathbb{P}(Y = j|X) = \frac{\mathbb{P}(X|Y = j) \omega_j}{\sum_{i=1}^k \mathbb{P}(X|Y = i) \omega_i}$$

and assign X to the topic that maximizes that probability.

The following theorem is a variation of Propositions 3 and 4 in Zou et al. (2013) and relates the observable moments of the known variables with the unknowns (M, Ω) . We will provide three estimators: $\tilde{M}_1 \in \mathbb{R}^n$, $\tilde{M}_2 \in \mathbb{R}^{n \times n}$ and $\tilde{M}_3 \in \mathbb{R}^{n \times n \times n}$ converging to the symmetric low rank tensors that will be used to retrieve the model parameters.

Theorem 2.1. *Fix a value $N \in \mathbb{N}$, and let $X^{(1)}, \dots, X^{(N)}$ be N sample documents generated according to a single topic model with parameters (M, Ω) . The following relations hold:*

- Define the vector $\tilde{M}_1 \in \mathbb{R}^n$, such that for each $h \in \{1, \dots, n\}$:

$$(\tilde{M}_1)_h := \frac{\sum_{i=1}^N (X^{(i)})_h}{\sum_{i=1}^N c_i},$$

then

$$\mathbb{E}[(\tilde{M}_1)_h] = \sum_{j=1}^k \omega_j \mu_{h,j}.$$

- Define the matrix $\tilde{M}_2 \in \mathbb{R}^{n \times n}$ such that, for each $h \neq l \in \{1, \dots, n\}$:

$$(\tilde{M}_2)_{h,l} := \frac{\sum_{i=1}^N (X^{(i)})_h (X^{(i)})_l}{\sum_{i=1}^N (c_i - 1) c_i}$$

$$(\tilde{M}_2)_{h,h} := \frac{\sum_{i=1}^N (X^{(i)})_h ((X^{(i)})_h - 1)}{\sum_{i=1}^N (c_i - 1) c_i};$$

then, for each $h, l \in \{1, \dots, n\}$,

$$\mathbb{E}[(\tilde{M}_2)_{h,l}] = \sum_{j=1}^k \omega_j \mu_{h,j} \mu_{l,j}.$$

- Define the tensor $\tilde{M}_3 \in \mathbb{R}^{n \times n \times n}$ such that, $h \neq l \neq m \in \{1, \dots, n\}$:

$$(\tilde{M}_3)_{h,l,m} := \frac{\sum_{i=1}^N (X^{(i)})_h (X^{(i)})_l (X^{(i)})_m}{\sum_{i=1}^N (c_i - 2)(c_i - 1) c_i}$$

$$(\tilde{M}_3)_{h,l,l} := \frac{\sum_{i=1}^N (X^{(i)})_h (X^{(i)})_l ((X^{(i)})_l - 1)}{\sum_{i=1}^N (c_i - 2)(c_i - 1) c_i}$$

$$(\tilde{M}_3)_{l,l,l} := \frac{\sum_{i=1}^N ((X^{(i)})_l ((X^{(i)})_l - 1) ((X^{(i)})_l - 2))}{\sum_{i=1}^N (c_i - 2)(c_i - 1) c_i};$$

then, for each $h, l, m \in \{1, \dots, n\}$,

$$\mathbb{E}[(\tilde{M}_3)_{h,l,m}] = \sum_{j=1}^k \omega_j \mu_{h,j} \mu_{l,j} \mu_{m,j}.$$

Given a sample, we are able to calculate the three estimators \tilde{M}_1 , \tilde{M}_2 and \tilde{M}_3 . If we look at their expected values, whose variables are unknown, we can notice that they have a form that is highly similar to matrix products and tensor multiplications. In particular, we can express those expectations in a more synthetic form, defining the first, second and third order tensors retrieved from the observable data

$$M_1 := \sum_{i=1}^k \omega_i \mu_i = M \Omega \tag{1}$$

$$M_2 := \sum_{i=1}^k \omega_i \mu_i \otimes \mu_i = \left(\sum_{j=1}^k \omega_j \mu_{h,j} \mu_{l,j} \right)_{h,l} = M \text{diag}(\Omega) M' \tag{2}$$

$$M_3 := \sum_{i=1}^k \omega_i \mu_i \otimes \mu_i \otimes \mu_i = \left(\sum_{j=1}^k \omega_j \mu_{h,j} \mu_{l,j} \mu_{m,j} \right)_{h,l,m} \tag{3}$$

We use the tensor notation to be in line with the cited literature; here, if v_1, \dots, v_r are vectors in \mathbb{R}^m , $v_1 \otimes \dots \otimes v_r$ is the m -dimensional vector such that $(v_1 \otimes \dots \otimes v_r)_{i_1, \dots, i_m} = (v_1)_{i_1} \dots (v_r)_{i_m}$.

Theorem 2.1 allows to express observable moments in the form of a symmetric tensor. By construction it is immediate to see that both M_2 and M_3 have symmetric-rank less than or equal to k , and the following simple limit holds, for $i = 1, 2, 3$:

$$\lim_{N \rightarrow \infty} \tilde{M}_i = M_i$$

We now provide a result that describes how fast this limit converges.

Theorem 2.2. *Let \tilde{M}_2 and \tilde{M}_3 the empirical estimates of M_2 and M_3 obtained using theorem 2.1; let also*

$$C_1 = \sum_{i=1}^N c_i, \quad C_2 = \sum_{i=1}^N (c_i - 1)c_i, \quad C_3 = \sum_{i=1}^N (c_i - 2)(c_i - 1)c_i$$

$$D_2 = \frac{(\sum_{i=1}^N (c_i(c_i - 1))^2 - C_2)}{C_2^2}, \quad D_3 = \frac{(\sum_{i=1}^N (c_i(c_i - 1)(c_i - 2))^2 - C_3)}{C_3^2}$$

and define also the operators $W_2 \in \mathbb{R}^{n \times n}$ and $W_3 \in \mathbb{R}^{n \times n \times n}$ as

$$(W_2)_{u,v} = \sqrt{\sum_{j=1}^k \mu_{u,j}^2 \mu_{v,j}^2} \omega_k, \quad (W_3)_{u,v,w} = \sqrt{\sum_{j=1}^k \mu_{u,j}^2 \mu_{v,j}^2 \mu_{w,j}^2} \omega_k.$$

then, for any $0 \leq \delta < 1$, we have that

$$\mathbb{P}(\|M_2 - \tilde{M}_2\|_F < \epsilon) > 1 - \delta$$

holds for any corpus whose document lengths (c_1, \dots, c_N) satisfy

$$\sqrt{\frac{1}{C_2} + \|W_2\|_F^2 D_2 - \frac{1}{N} \|M_2\|_F^2} + \sqrt{\log\left(\frac{1}{\delta}\right) \frac{\max_j (c_j) \sqrt{C_1}}{C_2}} < \epsilon$$

Also, for any pair $\epsilon > 0$ and $\delta > 0$, we have that

$$P(\|M_3 - \tilde{M}_3\|_F < \epsilon) > 1 - \delta$$

holds when

$$\sqrt{\frac{1}{C_3} + \|W_3\|_F^2 D_3 - \frac{1}{N} \|M_3\|_F^2} + \sqrt{\log\left(\frac{1}{\delta}\right) \frac{\max_j (c_j(c_j - 1)) \sqrt{C_1}}{C_3}} < \epsilon$$

Remark 2. We briefly comment the results of the theorem. We focus on M_2 (similar arguments holds for M_3), analyzing the case where all the documents have the same length c (so, for all i , $c_i = c$) and c is somewhat large (so $c \simeq c - 1$). Then the bound simplifies to:

$$\sqrt{\frac{1}{Nc^2} + \|W_2\|_F^2 \frac{1}{N} - \frac{1}{N} \|M_2\|_F^2} + \sqrt{\frac{1}{Nc} \log\left(\frac{1}{\delta}\right)}.$$

It is interesting to notice that the worst-case accuracy of the bound is $\epsilon = O(1/\sqrt{N})$. In general we can only say that $\|M_2\|_F^2 < \|M_2\|_1 = 1$ and $\|W_2\|_F^2 < 1$ but in our experiments $\|W_2\|_F$ and $\|M_2\|_F$ are usually far smaller. Also, the bound becomes smaller as c is large, with a clear limitation: if we have very few documents (N small), even if they are very long (large c), it is impossible to accurately learn the model, as in particular we may not even see all the topics.

Remark 3 (Alternative way of obtaining the formulation above). The formulation of theorem 2.1, which derives from (Zou et al., 2013) has a sample complexity that depends both on the documents length and on the size of the corpus we want to analyze. Alternative techniques can be found in literature to obtain from a text corpus described as in this section a symmetric low-rank tensor expression. The most popular is the one described in Anandkumar et al. (2012a), that, for each document $i \in \{1, \dots, N\}$ considers three words (selected randomly):

$$x_1^{(i)}, \quad x_2^{(i)}, \quad x_3^{(i)}$$

and then shows that

$$\begin{aligned} \frac{\sum_{i=1}^N (x_1^{(i)})_h}{N} &\xrightarrow{N \rightarrow \infty} (M_1)_h \\ \frac{\sum_{i=1}^N (x_1^{(i)})_h (x_2^{(i)})_l}{N} &\xrightarrow{N \rightarrow \infty} (M_2)_{h,l} \\ \frac{\sum_{i=1}^N (x_1^{(i)})_h (x_2^{(i)})_l (x_3^{(i)})_m}{N} &\xrightarrow{N \rightarrow \infty} (M_3)_{h,l,m} \end{aligned}$$

This method is highly unstable when dealing with small corpora, as uses only a small part of the available information (just three words for each document). In order to make it usable when the number of documents is small we would need either to reiterate the process many times or to "break" each document into many sub-documents with three words each; a pre-processing activity that is not needed in the technique of theorem 2.1. Also, it easy to see that the convergence bounds for these estimators are exactly the same of the ones presented in Theorem 2.2, with the difference that $c_i = 3$ for any document; to reach an arbitrary accuracy ϵ with high probability, we need $O(1/\epsilon^2)$ documents.

3 Latent Dirichlet Allocation

The obvious criticism of the single topic model is that each document can deal with a unique topic, an hypothesis that is commonly considered unrealistic. To overcome this issue, more complex models have been introduced, and one of these is Latent Dirichlet Allocation (LDA) (Griffiths and Steyvers, 2004; Blei et al., 2003). In its simplest form, LDA assumes that each document deals with a multitude of topics, in proportions that are governed by the outcome of a Dirichlet distribution. More precisely, considering our text corpus with N documents with a vocabulary of n words, the generative process for each text is the following:

- First a vector of topic proportions is drawn from a Dirichlet distribution with parameter $\alpha \in \mathbb{R}_+^k$, $Dir(\alpha)$; we recall that Dirichlet distribution is distributed over the simplex

$$\Delta^{k-1} = \{v \in \mathbb{R}^k : \forall i, v_i \in [0, 1], \text{ and } \sum v_i = 1\}$$

and has the following density function, for $h \in \Delta^{k-1}$:

$$\mathbb{P}(h) = \frac{\Gamma(\alpha_0) \prod_{i=1}^k h_i^{\alpha_i-1}}{\prod_{i=1}^k \Gamma(\alpha_i)}$$

Where $\alpha_0 = \sum \alpha_i$. From a practical point of view, this step consists in drawing a vector of parameters $h \in \Delta^{k-1}$ such that h_i represents the proportion of the topic i in the document.

- Once the topic proportions (also named *mixture of topics*) have been designed, each word of the document is generated according to the following procedure: first a (hidden) topic of the word, say $Y \in \{1, \dots, k\}$, is drawn, according to the probabilities defined by h (so we will have probability h_j of drawing topic j) and then we will generate the word itself according to a multinomial distribution; for each $i \in \{1, \dots, n\}$, $\mu_{i,j}$ will be the probability of generating the word i under the topic j :

$$\mathbb{P}(\text{Drawing word } i | Y = j) = \mu_{i,j}, \text{ and } M = (\mu_{i,j})_{i,j} \in \mathbb{R}^{n \times k}.$$

Again, we will denote with μ_i the set of columns of M :

$$M = [\mu_1 | \dots | \mu_k]$$

Maintaining the notation of the previous section, we will indicate $x_j^{(i)} \in \mathbb{R}^n$ as the coordinate vector indicating the word at position j in document i , $X^{(i)} = \sum x_i$ as the word-count vector of document i and $c_i = \sum_{j=1}^n (X^{(i)})_j$ as the number of words in that document. In the case of LDA, the unknown model parameters are M and α , so the pair (M, α) .

As in the case of the single topic model, we want to manipulate the observable moments in order to obtain a set of symmetric low rank tensors expressible as a product of the unknown parameters (as in eq. (1), (2) and (3)), in order to decompose those tensors and retrieve the parameters. The following theorem is an immediate modification of the one presented in (Anandkumar et al., 2012b, Lemma 3.2), and relates the observable moments of the known variables with the unknowns (M, α) ; providing the required representation. The only modification consists in the fact that we have used the estimates of Theorem 2.1 instead of the standard ones of Remark 3.

Theorem 3.1. *Let \tilde{M}_1, \tilde{M}_2 and \tilde{M}_3 the empirical estimates defined in Theorem 2.1. Define*

$$\tilde{M}_2^\alpha := \tilde{M}_2 - \frac{\alpha_0}{\alpha_0 + 1} \tilde{M}_1 \otimes \tilde{M}_1$$

$$\tilde{M}_3^\alpha := \tilde{M}_3 - \frac{\alpha_0}{\alpha_0 + 2} (M_{1,2}) + \frac{2\alpha_0^2}{(\alpha_0 + 2)(\alpha_0 + 1)} \tilde{M}_1 \otimes \tilde{M}_1 \otimes \tilde{M}_1$$

where $M_{1,2} \in \mathbb{R}^{n \times n \times n}$ is a three dimensional tensor such that

$$(M_{1,2})_{h,l,m} = ((\tilde{M}_2)_{h,l}(\tilde{M}_1)_m + (\tilde{M}_2)_{l,m}(\tilde{M}_1)_h + (\tilde{M}_2)_{m,h}(\tilde{M}_1)_l)$$

Then

$$\mathbb{E}[\tilde{M}_2^\alpha] = \sum_{i=1}^k \frac{\alpha_i}{(\alpha_0 + 1)\alpha_0} \mu_i \otimes \mu_i = M_2^\alpha$$

$$\mathbb{E}[\tilde{M}_3^\alpha] = \sum_{i=1}^k \frac{2\alpha_i}{(\alpha_0 + 2)(\alpha_0 + 1)\alpha_0} \mu_i \otimes \mu_i \otimes \mu_i = M_3^\alpha$$

This technique allows to express observable moments in the form of a symmetric tensor. Both M_2^α and M_3^α have symmetric-rank less than or equal to k , and so we can use any tensor decomposition algorithm to retrieve the unknown model parameters (M, α) from them. A major advantage of this theorem, as of the homologous theorem in Anandkumar et al. (2012b), is that it only requires the knowledge of the value α_0 , while non-spectral methods require the knowledge of the full vector α .

Remark 4 (Inference). Similarly to the single topic model, one of the main usages of LDA is to infer the mixture of hidden topics of each document in a corpus. Unfortunately, an exact formula to perform this inference is not known, but a number of approximate approaches exist, like Gibbs sampling (Griffiths and Steyvers, 2004; Newman et al., 2009) and Expectation Propagation (Blei et al., 2003). In our case, if we assume to know the values of model parameters (M, α) , we can apply a modified Gibbs Sampling to infer the topic mixture for a given text; consider a text, whose words are x_1, \dots, x_c ; then, in LDA, each word x_i is generated by a unique topic Y_{x_i} . Using the equations for Gibbs Sampling from (Griffiths and Steyvers, 2004), if Y_{x_i} is the hidden topic of word x_i and Y_{-x_i} is the set of topic assignment for all the words in the document excluded x_i , it can be shown that

$$\mathbb{P}(Y_{x_i} = j | Y_{-x_i}, x_i) \approx \mu_{x_i, j} \frac{n_{-i, j} + \alpha_j}{c - 1 + \alpha_0} \quad (4)$$

where $n_{-i, j}$ is the number of words assigned to topic j excluding x_i , c is the total number of words in the document and $\mu_{x_i, j}$ is the probability of drawing the word x_i under topic j . So, given a document, first we have to assign to each word a hidden topic, and then update this assignment word by word in a iterative way, using a monte-carlo assignment governed by equation (4). Each iteration updates the number of words assigned to a given topic; after a suitable number of iterations, we can estimate the topic mixture for a given document as the vector $h \in \mathbb{R}^k$ such that

$$(h)_j = \frac{n_j + \alpha_j}{c + \alpha_0}$$

where n_j is the number of words assigned to topic j .

4 The Core Algorithm

We now present the algorithm to retrieve the parameters of a LVM, once a symmetric tensor expression like the ones in equations (1), (2) and (3) are provided. We will use here the notation of Section 2, focusing on the single topic model, as the extension to LDA and to other LVM is straightforward (see Anandkumar et al., 2014). The core of our algorithm consists in retrieving the values of the unknowns in equations (1),(2) and (3) by first getting from them a three dimensional tensor H in $\mathbb{R}^{n \times k \times k}$ and then performing n *SVD* on the fibers of H (belonging to $\mathbb{R}^{k \times k}$) obtaining the required unknowns as the singular values of that fibers; for this reason we name our method *SVTD, Singular Value based Tensor Decomposition*. We want our method to work only on with standard matrix operations, and to accomplish this need, we select a feature (in the text mining example, a word) r , among the n available and, instead of considering the full tensor M_3 , we work only with its r -th fiber. This is made explicit in the following definition.

Definition 4.1 (Notation). *Given any $r \in \{1, \dots, n\}$, we define:*

$$M_r := (\mu_{i, j})_{i, j: i \neq r} \in \mathbb{R}^{(n-1) \times k},$$

$$M_{2, r} := M_r \text{diag}(\Omega) M_r' = \left(\sum_{j=1}^k \mu_{l, j} \mu_{h, j} \omega_j \right)_{l, h \neq r} \quad (5)$$

$$M_{3, r} := M_r \text{diag}(\Omega) \text{diag}(\mu_r) M_r' = \left(\sum_{j=1}^k \mu_{l, j} \mu_{h, j} \mu_{r, j} \omega_j \right)_{l, h \neq r} \quad (6)$$

We first observe that both matrices $M_{2,r}$ and $M_{3,r}$ are in $\mathbb{R}^{(n-1) \times (n-1)}$ and have low rank $\leq k < n$. Matrix M_r is M after removing the r -th row, as well as $M_{3,r}$ is the r -th slice of the three dimensional tensor M_3 after the removal of feature r .

It is easy to see that, in the case of the single topic model, all the entries of $M_{2,r}$ and $M_{3,r}$ are easily estimable via the empirical formulas of Theorem 2.1, while Theorem 3.1 provides us the estimates for LDA. Also for more complex LVM, we can estimate those values using the techniques outlined in (Anandkumar et al., 2014).

We now introduce our algorithm (SVTD), whose key steps are outlined in Algorithm (1); for a given r , when M_1 , $M_{2,r}$ and $M_{3,r}$ are provided, this technique is able to retrieve the values of the hidden parameters (M, Ω) in few simple steps.

Algorithm 1 Complete algorithm - SVTD

Require: M_1, M_2, M_3 positive semidefinite with rank k

- 1: Decompose M_2 as $M_2 = EE'$, where $E \in \mathbb{R}^{n \times k}$ with rank k
 - 2: Select a feature r and compute $M_{3,r}$
 - 3: Compute $E_r \in \mathbb{R}^{(n-1) \times k}$ removing the r -th row from E
 - 4: Find O and μ_r with a SVD on $H_r := E_r^* M_{3,r} (E_r')^*$
 - 5: **for** $i = 1 \rightarrow n, i \neq r$ **do**
 - 6: Compute E_i removing the i -th row from E and get $H_i := E_i^* M_{3,i} (E_i')^*$
 - 7: Obtain μ_i as the diagonal entries of $O' H_i O$
 - 8: **end for**
 - 9: Obtain Ω solving $M_1 = M\Omega$
 - 10: **return** (M, Ω)
-

The constructive proof of the following theorem will explain why SVTD performs a correct retrieval of the desired model parameters.

Theorem 4.1. *If all the elements of μ_r are distinct and $M_{2,r}$ and $M_{3,r}$ have rank k , then SVTD produces exactly the values of (M, Ω) .*

Proof. Given M_2 we can decompose it as

$$M_2 = EE'$$

where E is a rank- k matrix in $\mathbb{R}^{n \times k}$. If now we consider $M_{2,r}$, we can easily show that

$$M_{2,r} = E_r E_r'$$

where E_r is a rank- k matrix in $\mathbb{R}^{(n-1) \times k}$ obtained by removing the r -th column from E . This decomposition is unique up to an isometry of \mathbb{R}^k ; this means that there exists an orthogonal matrix $O \in \mathbb{R}^{k \times k}$ such that

$$M_{2,r} = M_r(\text{diag}(\Omega))M_r' = E_r O O' E_r'$$

and so

$$E_r O = M_r(\text{diag}(\Omega))^{\frac{1}{2}}. \tag{7}$$

We now look for that isometry, exploiting the matrix $M_{3,r}$. By construction we have

$$M_{3,r} = M_r \text{diag}(\Omega) \text{diag}(\mu_r) M_r' = E_r O \text{diag}(\mu_r) O' E_r'. \tag{8}$$

So it holds

$$H_r := E_r^* M_{3,r} (E_r')^* = O \text{diag}(\mu_r) O' \quad (9)$$

where $E_r^* = (E_r' E_r)^{-1} E_r'$ is the Moore–Penrose pseudoinverse of E_r . As all the elements of μ_r are distinct, the decomposition at the right side of the equation is unique up to a reordering of the columns of O and possible change of sign. O can be obtained just by performing a SVD on $E_r^* M_{3,r} (E_r')^*$, and then used to diagonalize it as follows:

$$O' H_r O = \text{diag}(\mu_r). \quad (10)$$

In this way we retrieve the vector μ_r and the matrix O . As Lemma 4.1 says, the matrix O does not depend on the feature we selected for its construction; this means that if we would have chosen a different feature r , the matrix O would have come out to be exactly the same up to a column reordering. So, we arbitrary choose the column reordering obtained by isolating the feature r and we obtain the values of the vectors μ_i , for any other $i \neq r$, just by calculating

$$H_i := E_i^* M_{3,i} (E_i')^* \quad (11)$$

and getting μ_i from

$$O' H_i O = \text{diag}(\mu_i).$$

with the same O used for the feature r . Iterating on the various features $i \in \{1, \dots, n\}$ we obtain the matrix M . The subsequent estimations of Ω is straightforward, and can be obtained by solving the linear system $M_1 = M\Omega$. \square

As can be seen from the proof, the method just consists in three logical steps: first, we retrieve the matrix $O \in \mathbb{R}^{k \times k}$, then we get tensor $H \in \mathbb{R}^{n \times k \times k}$ whose i -th fiber is $H_i \in \mathbb{R}^{k \times k}$ defined as in (11) and as a last step we get the values of the rows of M as the diagonal elements of $O' H_i O$.

Lemma 4.1. *Let $E \in \mathbb{R}^{n \times k}$ with rank k satisfying $M_2 = EE'$, and $E_r \in \mathbb{R}^{(n-1) \times k}$ as E with the r -th row removed. Then, there exist two isometries O_r and O realizing the equations*

$$E_r O_r = M_r (\text{diag}(\Omega))^{\frac{1}{2}}, \quad E O = M (\text{diag}(\Omega))^{\frac{1}{2}}$$

and it holds $O = O_r$.

Remark 5 (On the generality of the algorithm). We remark that during the construction of the algorithm we have not made any hypotheses on the distribution of the data. In fact, we can state that the presented algorithm just decomposes a set of symmetric tensors, obtaining as output the unknown parameters of the model that underlies the data. A consequence is that SVTD can be used to learn efficiently many kind of latent variable models: Gaussian mixtures, Latent Dirichlet allocation, Hidden Markov model, and all the models described in (Anandkumar et al., 2014). In this sense, this algorithm is just a new alternative to the tensor decomposition method described in that work, or to the methods presented in (Anandkumar et al., 2012a) or (Anandkumar et al., 2012b). In the experiments section we will compare the ability of this method to learn the model parameters, compared with current state of the art algorithms.

Remark 6 (Alternative algorithms). As said in the introduction, other algorithms exists to retrieve the unknowns (M, Ω) from M_1, M_2 and M_3 . The most popular and solid way is the one described in (Anandkumar et al., 2014); there, an iterative algorithm is used to decompose M_3 and obtain the

desired unknowns. This algorithm is the three-dimensional extension of the well-known matrix power method to obtain eigenvectors of a matrix; while very robust, its implementation is complex and requires many lines of code; in addition, it is an iterative method, and so it requires a tuning of the hyperconvergence parameters, that might require many trial-and-error tests. In (Anandkumar et al., 2012b) an algorithm is presented that, while similar to the one presented here, relies on the singular *vectors* of a SVD decomposition, while our deals with the singular *values*, that are known to be more stable under perturbations (see Stewart, 1998 for perturbation results on singular values and singular vectors). The algorithm presented in (Anandkumar et al., 2012a) is also different: it generates a linear asymmetric operator from data and then works with its eigenvalues. Both the methods in (Anandkumar et al., 2012a) and (Anandkumar et al., 2012b) need randomized matrices to work, and this, in our experiments makes the results slightly unstable, depending on the randomized matrix. We believe that SVTD might be a viable alternative: it seems very robust in the experiments, and can be implemented in any language in just few lines of code. Furthermore it is not an iterative method, so no hyperparameter tuning is needed and the running time is deterministic.

Remark 7 (On Finding E). There exists a straightforward way to calculate the matrix E : as matrix M_2 is a symmetric positive semidefinite rank- k matrix, it has a SVD:

$$M_2 = USU' = U_k S_k U_k'$$

where U_k and S_k are the matrices of singular vectors and values truncated at the k -th smallest eigenvalues (the smallest non-zero eigenvalue). Then, we can easily find the matrix

$$E = U_k S_k^{\frac{1}{2}} \in \mathbb{R}^{n \times k}.$$

Remark 8 (On the selection of feature r). The initial steps of the algorithm need the isolation of a feature r to compute the matrix O . While theoretically we could select any feature r such that all elements of μ_r are distinct, it is clear that, if matrices M_1 , M_2 and M_3 are subject to perturbations, the results obtained by the algorithm might vary a lot, depending on the selected feature r . Theorem 5.1 will show that the accuracy of the algorithm under perturbed data, will depend on how different are the elements in μ_r . A consequence of this is that a good way to find feature r , and so a reliable matrix O , might be to repeat the first four steps of the algorithm isolating different features and select the one that maximizes the quantity

$$\min_{i \neq j} (|\mu_{i,r} - \mu_{j,i}|)$$

Remark 9 (Complexity analysis). We start analyzing the time complexity. Using randomized SVD techniques (see Halko et al., 2011), step 1 can be carried out with a total of $O(n^2k)$ steps. Step 4 also requires $O(n^2k)$ steps for matrix multiplication, $O(nk^2)$ steps for pseudo-inverse computation, while the SVD on $E_r^* M_{3,r} (E_r')^*$ requires $O(k^3)$ steps. The operations in the loop of steps 6 and 7 just require $O(nk^2)$ steps for the calculation of the pseudo-inverse E_r^* and $O(n^2k)$ for the matrix multiplication of step 7; so the loop has total complexity $O(n^3k + n^2k^2)$.

The overall complexity of the algorithm is thus

$$O(n^3k + n^2k^2 + k^3).$$

Nevertheless, it is important to highlight that the implementation described in Algorithm 1 has mainly a descriptive purpose, while for a specific LVM, optimized implementations can be done;

for the single topic model, for example, the method can be implemented without never calculating explicitly the tensor M_3 , calculating in one step the tensor H , with a complexity of $O(Nnk)$ and then performing the subsequent diagonalizations in $O(n^2k^2)$ time. Additional tuning of the performances can be obtained exploiting the sparsity of X , using for matrix operations sparse matrix technique. We also remark that the algorithm is trivially parallelizable: assuming we have m machines on which to parallelize the steps 5, 6, 7 of the algorithm, we can reduce the total running time to

$$O\left(\frac{n^3k + n^2k^2}{m} + k^3\right).$$

Regarding memory, notice that we never use the full tensor M_3 , but only its $r - th$ slice; the overall memory complexity of the algorithm is thus $O(n^2)$.

These complexity requirements are comparable to the ones of Anandkumar et al. (2014, 2012a,b); however, these methods have the drawbacks already mentioned: the tensor power method in Anandkumar et al. (2014) is an iterative technique, with a number of iterations difficult to bound a priori; the authors suggest that accuracy ϵ can be reached with $O(k^{5+\delta}(\log(k) + \log\log(1/\epsilon)))$ operations, among iterations, random restarts and actual matrix operations, compared to our $O(k^3)$; to this time we need to add the time necessary to get the a $k \times k \times k$ tensor from the sample. The techniques in (Anandkumar et al., 2012a) and (Anandkumar et al., 2012b) are randomized, with nontrivial variance in their output, so they may require several runs of the full algorithm in order to provide accurate results.

5 Perturbation Analysis

In the previous section we have outlined an algorithm that accurately learns the model parameters of a LVM, given the exact values of M_1 , M_2 and M_3 . However, when applying our algorithm to a real world problem, we never have these exact variables, but only a set of estimators \tilde{M}_1 , \tilde{M}_2 and \tilde{M}_3 that are expected to become arbitrarily accurate as sample size increases. This fact has some immediate consequences: first we need to adapt SVTD to deal with perturbed matrices (as \tilde{M}_1 , \tilde{M}_2 and \tilde{M}_3 are) that do not necessarily have rank k or are not assured to be positive definite. Second, we need to study how the perturbations on those estimates propagate up to the final results. The adaptation of SVTD is outlined in Algorithm 2.

The modifications with respect to Algorithm 1 need to guarantee that we deal with positive definite matrices with rank k . We do this by defining \tilde{E} , in steps 1 and 2, as the product of the first k left singular vectors of \tilde{M}_2 and the first k singular values and, in step 5, taking \tilde{O} to be the left singular vectors of $\tilde{M}_{3,r}$; the rest of the algorithm is identical.

Remark 10. In step 1 of Algorithm 2 we obtain

$$\tilde{M}_2 = \tilde{U}\tilde{S}\tilde{V}.$$

In general, as \tilde{M}_2 converges to M_2 which is positive semidefinite, we expect that for suitably big samples $V = U'$, i.e. that \tilde{M}_2 is also positive semidefinite. The size of the sample required to have \tilde{M}_2 positive semidefinite depends on the concentration properties of the sample. However, Algorithm 2 produces accurate results even when positive definiteness of \tilde{M}_2 is not guaranteed, as explained next in Theorem 5.1.

A python implementation of this algorithm can be found at this link <https://github.com/mruffini/SpectralMethod.git>.

Algorithm 2 SVTD when \tilde{M}_1, \tilde{M}_2 and \tilde{M}_3 are available, instead of M_1, M_2 and M_3 .

Require: $\tilde{M}_1, \tilde{M}_2, \tilde{M}_3$, rank k

- 1: Perform an SVD on \tilde{M}_2 , obtaining $\tilde{M}_2 = \tilde{U}\tilde{S}\tilde{V}$.
 - 2: Obtain $\tilde{E} \in \mathbb{R}^{n \times k}$ as $\tilde{E} = \tilde{U}_k \tilde{S}_k^{\frac{1}{2}}$, where \tilde{U}_k and \tilde{S}_k are \tilde{U} and \tilde{S} truncated at the k -th singular vector.
 - 3: Select a feature r and compute $\tilde{M}_{3,r}$
 - 4: Compute $\tilde{E}_r \in \mathbb{R}^{n-1 \times k}$ removing the r -th row from \tilde{E}
 - 5: Find \tilde{O} and $\tilde{\mu}_r$ as the left singular vectors and values of $\tilde{H}_r := \tilde{E}_r^* \tilde{M}_{3,r} (\tilde{E}_r')^*$
 - 6: **for** $i = 1 \rightarrow n, i \neq r$ **do**
 - 7: Compute \tilde{E}_i removing the i -th row from \tilde{E} and get $\tilde{H}_i := \tilde{E}_i^* \tilde{M}_{3,i} (\tilde{E}_i')^*$
 - 8: Obtain $\tilde{\mu}_i$ as the diagonal entries of $\tilde{O}' \tilde{H}_i \tilde{O}$
 - 9: **end for**
 - 10: Obtain $\tilde{\Omega}$ solving $\tilde{M}_1 = M \tilde{\Omega}$
 - 11: **return** $(\tilde{M}, \tilde{\Omega})$
-

We now study the accuracy of the algorithm. Intuitively, the more similar the perturbed \tilde{M}_1, \tilde{M}_2 and \tilde{M}_3 are to the exact M_1, M_2 and M_3 , the better the outcomes of the algorithm should be. This intuition is confirmed by the following theorem.

Theorem 5.1. *Given the unperturbed versions of M_1, M_2 and M_3 and the feature we want to isolate, r , let α and α_M be*

$$\alpha = \min_{i \neq j} (|\mu_{i,r} - \mu_{j,r}|), \quad \alpha_M = \min_{i \neq j} (|\sigma_i(M_2) - \sigma_j(M_2)|)$$

where $\sigma_i(M_2)$ are the singular values of M_2 . Assume the empirical estimates \tilde{M}_2 and \tilde{M}_3 satisfy

$$\|\tilde{M}_2 - M_2\|_F < \epsilon, \quad \|\tilde{M}_3 - M_3\|_F < \epsilon.$$

Then Algorithm (2), fed with \tilde{M}_1, \tilde{M}_2 and \tilde{M}_3 , provides an estimated matrix

$$\tilde{M} = [\tilde{\mu}_1 | \dots | \tilde{\mu}_k]$$

such that, for all i

$$\|\mu_i - \tilde{\mu}_i\|_2 \leq \frac{\epsilon}{\alpha} (C_1 + \frac{C_2}{\alpha_M}) + O(\epsilon^2)$$

where C_1 and C_2 are polynomial functions of $\|E_i\|_F, \|O\|_F$ and $\|M_{3,i}\|_F$.

Note the key role of α , the minimum difference between the elements of μ_r ; when samples are large enough, the theorem guarantees that the algorithm works correctly, although “large enough” depends on α . When this condition is not satisfied, the learning algorithm still works, but might provide output results that are different from the theoretical generative model. We recall here Remark 8, where we wondered how to select the proper feature r ; ideally, the one that would guarantee the highest accuracy would be the one with the highest possible α . Also there is a dependence on α_M . We suspect that these dependencies can be removed, improving this theorem, as the results are not expected to depend on this element.

6 Experiments

In this section we will provide some experiments, to test both on synthetic and real data, the algorithm we presented in this paper.

6.1 Synthetic data

6.1.1 Recovering M_2 and M_3

In Section 2 we described a technique derived from Zou et al. (2013) to recover the matrix M_2 and tensor M_3 from a sample, comparing them with the methods presented in the state of the art literature, outlined in Remark 3. In this section we compare, using synthetically generated data, how well the two different methods recover M_2 and M_3 as a function of the sample size. To perform this experiment, we generated a set of 1000 synthetic corpora according to the single topic model described in Section 2, with different sizes (the number of texts for each corpus); the smallest corpus contained 100 texts, the biggest 10000; each text was made of 10 words. For each corpus, the values of the unknowns (M, Ω) have been randomly generated, and from them we have been able to obtain the theoretical values of M_2 and M_3 using equations (2) and (3) and to compare those values with the one empirically estimated from data using the equations in Theorem 2.1 for the presented method and equations in Remark 3 for the competing one. Results appear in Figure 1, where we show how the estimated M_2 and M_3 , say \tilde{M}_2 and \tilde{M}_3 , approach the theoretical values; in particular, in the chart are represented the errors

$$Err_2 = \|\tilde{M}_2 - M_2\|_F, \quad Err_3 = \|\tilde{M}_3 - M_3\|_F$$

as a function of the sample size N used to find \tilde{M}_2 and \tilde{M}_3 . We can see that the method of Theorem 2.1 outperforms state of the art techniques, like the one presented in Remark 3.

6.1.2 Recovering (M, Ω) from a random sample

In this section we want to test the ability of the algorithm presented in this paper to recover the unknown parameters from random set of data. In particular, we fix a dictionary of $n = 100$ words with $k = 5$ topics and we proceed as in the previous section to generate the sample X : for various sizes comprehended between $N = 100$ and $N = 10000$ we generate synthetic corpora and we use them to learn the model parameters. For each sample corpus X we proceed as follows:

- We estimate the values of \tilde{M}_1 , \tilde{M}_2 and \tilde{M}_3 using Theorem 2.1.
- We retrieve from the estimated \tilde{M}_1 , \tilde{M}_2 and \tilde{M}_3 the pair of unknowns $(\tilde{M}, \tilde{\Omega})$ using SVTD as in Algorithm 2. Also we generate alternative solutions using the decomposition algorithms from Anandkumar et al. (2014) ("Tensor power method"), from Anandkumar et al. (2012a) ("Eigendecomposition method") and from Anandkumar et al. (2012b) ("SVD method"), that are the current reference methods.
- For each set of retrieved parameters $(\tilde{M}, \tilde{\Omega})$ we calculate the learning error as follows:

$$Err = \|\tilde{M}diag(\tilde{\Omega})\tilde{M}' - Mdiag(\Omega)M'\|_F$$

where (M, Ω) are the parameters used to generate the random sample corpus.

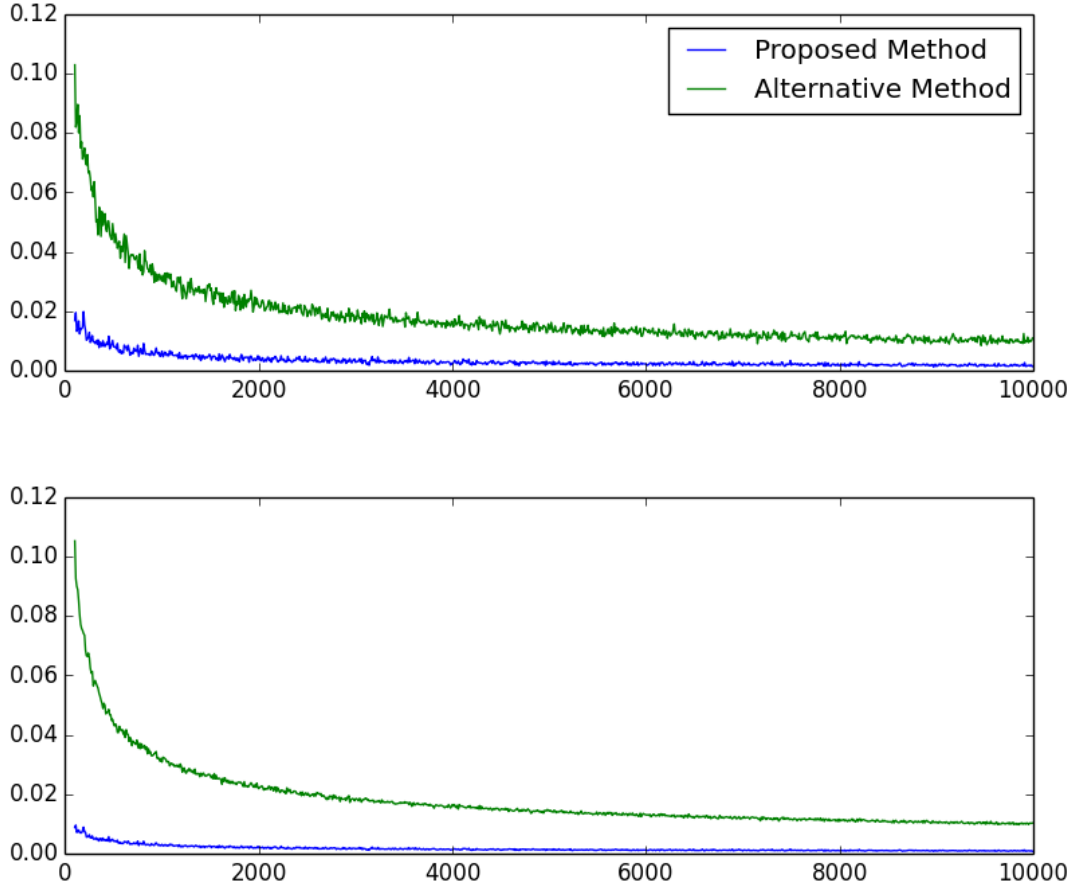


Figure 1: The x -axis of the figure represents the size of the synthetic text corpora, while the y -axis is Err_2 for the top chart and Err_3 for the bottom chart. Blue lines represent the errors obtained with the method presented in theorem 2.1, while green lines refer to the methods of Remark 3

- We plot in Figure 2 the results of the analysis as a function of N .

First of all we can see that all the methods perform in a similar way; peaks are present when a method provides results that are far from the latent variable used to generate the sample. We can see that the convergence of the presented SVTD seems to be smoother than that of the other techniques, as a consequence to the fact that the method is deterministic.

Running time and practical considerations. In linear algebra operations, the running time is highly influenced by many implementation details, such as the usage of vectorized operations. In our implementation, all the algorithms operate quite fast, requiring less than 5 seconds to recover from a sample of size $N = 10,000$, with a vocabulary of $n = 100$ words. All the algorithms have been implemented in Python 2.7, using *numpy* (Van Der Walt et al., 2011) library for linear algebra operations. All the experiments have run on a MacBook Pro, with an Intel Core i5 processor.

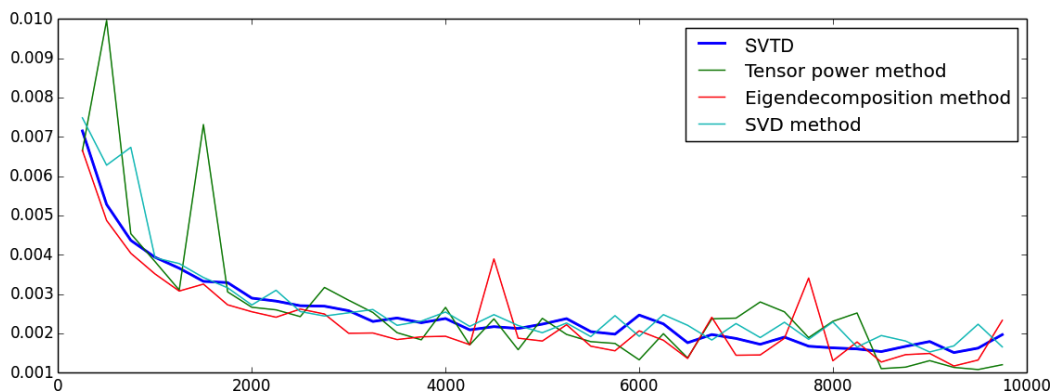


Figure 2: The x -axis of the figure represents the size of the synthetic text corpora, while the y -axis is Err for the various tested methods: the presented algorithm (blue line) behaves in a very similar way to tensor decomposition.

6.2 Real data

To perform experiments with real data, we analyzed two different corpora: the list of State of the Union addresses since 1945 to 2005 and Dante’s “Divina Commedia”. In both cases, we used a dictionary of $n = 3000$ words, so the data matrix X was a $N \times 3000$, where N was the size of the corpus (we had $N = 65$ on the first example and $N = 100$ in the second), while tensor M_3 belonged to $\mathbb{R}^{3000 \times 3000 \times 3000}$. We deliberately used corpora with $N \ll n$, in order to test the ability of our algorithm to work with small sets of documents.

6.2.1 Dante’s Divina Commedia

Dante’s “Divina Commedia” (Alighieri, 1979) is an Italian epic poem written in the first half of 14th century; it deals with the imaginary trip of the main character, Dante, in the afterlife, guided by Virgilio, the famous Latin poet, and Beatrice, a Florentine woman that inspired most of Dante’s works. The story line represents an allegorical description of death soul’s journey towards God according to medieval world view. It begins with Dante’s travel through the “Inferno” (Hell), where damned souls are deemed to eternal punishment according to their sins; the journey then moves to “Purgatorio”, a seven level mountain, where, at each level, a capital sin (sins less serious than those punished in Hell) is allegorically described; here souls are discounting their punishment, before finally move to “Paradiso”, Heaven, that is visited by Dante in the last third of the book. The book is made of 100 different chapters: 34 for Hell, 33 for Purgatory and 33 for Heaven.

Single Topic Model

We run SVTD for the Single Topic Model on the “Divina Commedia” corpus, on the $N = 100$ texts, represented by the 100 chapters of the book. We tested various possible number of topics, but surprisingly almost always the algorithm produced two significant topics. In the table 1 we can see the results of the algorithm run with $k = 2$: for each of the two topics a cluster has been defined; the most representative words are represented together with the chapters assigned to each cluster.

The full text can be found here: <http://www.gutenberg.org/files/1012/1012-0.txt>

We can see that most of the chapters of Hell are assigned to the same cluster; also the chapters of Heaven are all assigned to the same cluster while Purgatory is assigned in part to cluster 1 and in part to cluster 2.

Cluster	Most representative words	Assigned chapters
1	fosso, bolgia, scoglio, avante, coda, grido, rotta, ponte, stanchi, maestro	Inferno (1,3-10, 12-34) , Purgatorio (1-8, 10-12,20-24,26,27,29)
2	milizia, segue, intende, letizia, conosce, lumi, ama, Beatrice, piacer, cristo	Inferno (2,11) Paradiso (1-33), Purgatorio (6, 13-19, 25, 28, 30-33)

Table 1: The cluster assignment for “Divina Commedia” chapters.

Latent Dirichlet Allocation

As a second step, we used the same corpus, with the same vocabulary, to infer for each chapter how much it deals with topic 1 and how much it deals with topic 2, using Latent Dirichlet Allocation. In particular, we retrieved from the data the tensors \tilde{M}_2^α and \tilde{M}_3^α from Theorem 3.1 and we used them to feed Algorithm 2. We had to specify a value for α_0 , that was set to 2. With Algorithm 2 we retrieved a pair (M, α) . We then used this pair and partially collapsed Gibbs Sampling to infer the topic mixture for each chapter. As we just have two topics, it is easy to plot the results of that inference, see Figure 3. In this chart, the x -axis represents the progression of the chapters along the book, while red and blue lines represent the value of the topic proportions for each chapter. We can see that most of the first 34 chapters have a strong predominance of the first topic, marked with the red line, that consequently can be identified with the Hell topic. In the same way, second topic, or Heaven, is very strong in the last 33 chapters. Purgatory, in the middle part of the plot, has a mixed belonging. The amazing fact is that the proportion of the Heaven topic seems to increase as the chapters approach the Heaven section, corresponding to Dante’s ascent of the Purgatory mountain.

6.2.2 State of the Union addresses

Each year, the president of United States of America presents a speech to a joint session of the United States Congress, where he outlines his governative agenda, the national priorities and legislative projects. We considered the set of $N = 65$ state of the union addresses presented between 1945 and 2005, and we applied to this corpus the algorithm for the Single Topic Model, with the purpose of finding the most representative topics of the corpus and clustering the various speeches according to the learned topics. We run SVTD assuming to have $k = 5$ different topics, although with different values of k results were similar; we then grouped the speeches assigning them to the topic with the highest likelihood, using the Bayesian posterior assignment of Remark 1. In Table 2 we can see the results of this operation. For each topic a cluster has been defined; the most representative words are represented together with the speeches assigned to each cluster; for each president, the brackets indicate the year of the speech.

The full corpus can be found in link http://www.nltk.org/nltk_data.

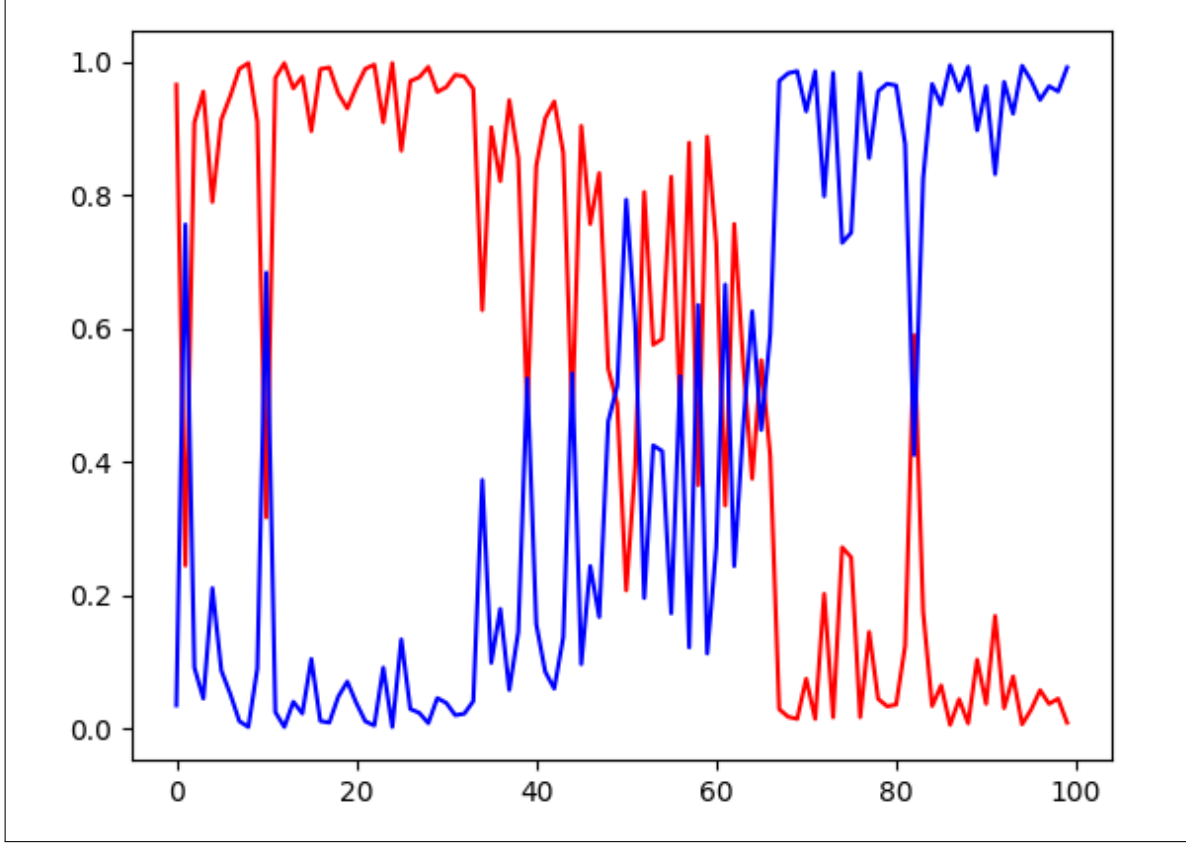


Figure 3: In the figure, the x -axis represents the chapter of the book: the first 34 are Hell, from 35 to 67 we have Purgatory and the last 33 are Heaven. The red line represents how much each chapter belongs to the first cluster, while the blue line how much it belongs to second cluster.

Cluster 1 contains only speeches of the Cold War period. Words like 'peacetime', 'construction' and 'projects' characterize, for instance, the period after the second World War, with the Marshall plan and Truman doctrine. Cluster 2 has speeches that belong to a wider set of dates, but all seem to be about internal politics and economics; among the most representative words we can find 'taxpayers', 'regulations' and 'productivity', key themes of the Reagan administration. Cluster 3 is clearly related to war with words as 'missile', 'conflict' and 'struggle'; the speeches come from presidents involved in important wars (WWII and Gulf wars). Cluster 4 mainly contains Bill Clinton speeches and has words that characterize the economic expansion of the 90ies ('companies', 'invest', 'credit'). Cluster 5 has among the top words 'terror', 'Afghanistan' and 'Iraq'; they all reveal post 9/11 politics carried on by G.W. Bush.

7 Conclusion and Future Work

We described a simple algorithm to learn latent variable models in polynomial time. A natural future work is to adapt this algorithm to an online, streaming environment (Liberty, 2013; Jain et al., 2016). In the theoretical front, we want to improve the perturbation Theorem 5.1, removing the

Cluster	Most representative words	Assigned speeches
1	construction, fiscal, legislative, peacetime, facilities, recommendations, projects, existing, transportation, veterans	Truman (1946 to 1950), Eisenhower (1953 to 1957, 1959), Kennedy (1961 to 1963), Johnson (1966,1967), Nixon (1973), Ford (1975,1977)
2	reducing, regulations, recovery, taxpayers, market, bills, weeks, nothing, gone, productivity	Johnson (1963 to 1965, 1968, 1969), Nixon (1970 to 1972, 1974), Ford (1976), Carter (1978), Reagan (1981 to 1988), Bush (1990, 1992)
3	ideals, soviet, missile, potential, missiles, world, conflict, struggle, countries, threat	Truman (1945, 1951), Eisenhower (1958, 1960), Carter (1979, 1980), Bush (1991a,1991b), G.W. Bush (2003)
4	companies, invest, 21st, teachers, parents, revolution, lowest, challenge, credit, bipartisan	Bush (1989), Clinton (1993 to 2000)
5	September, enemies, terror, compassion, terrorists, Afghanistan, relief, retirement, Iraq, dangerous	G.W. Bush (2001a, 2001b, 2002, 2004, 2005)

Table 2: The cluster assignment for the State of the Union addresses.

dependence from α and α_M . In the applications side, we are interested in applying this algorithm to learn LVM in the healthcare analytics field, for instance to construct disease projection models and disease clustering. Genetic data, where e.g. one typically has many more genes or SNPs than sequenced individuals, would also be of interest.

Acknowledgements

We are grateful to thank Daniel Hsu for his guidance on the spectral method for LDA. M. Casanellas is partially funded by AGAUR project 2014 SGR-634 and MINECO/FEDER project MTM2015-69135-P. R. Gavaldà is partially funded by AGAUR project 2014 SGR-890 (MACDA) and by MINECO project TIN2014-57226-P (APCOM).

Appendix

A Proofs for Section 2

A.1 Proof of Theorem 2.1

Proof. We will prove the statements only for

$$\mathbb{E}\left(\frac{\sum_{i=1}^N (X^{(i)})_h (X^{(i)})_l}{\sum_{i=1}^N (c_i - 1) c_i}\right) = \sum_{j=1}^k \omega_j \mu_{h,j} \mu_{l,j}.$$

Similar arguments hold for the other equations. It is easy to see, by conditional independence, that

$$E((X^{(i)})_h (X^{(i)})_l) = \sum_{j=1}^k \omega_j E((X^{(i)})_h (X^{(i)})_l | Y = j)$$

but the conditioned $(X^{(i)})_h$ and $(X^{(i)})_l$ are components of a multinomial distribution and so

$$\sum_{j=1}^k \omega_j E((X^{(i)})_h (X^{(i)})_l | Y = j) = \sum_{j=1}^k \omega_j (c_i^2 - c_i) \mu_{h,j} \mu_{l,j}$$

which implies the thesis. \square

A.2 Proof of Theorem 2.2

Proof. We want to express in a suitable way the elements of the matrix

$$\tilde{M}_2 - M_2 \tag{12}$$

and then express a bound using McDiarmid's inequality (McDiarmid, 1989). We know by construction that, for any $i \in \{1, \dots, N\}$ it holds that

$$X^{(i)} = \sum_{j=1}^{c_i} x_j^{(i)}$$

where each $x_j^{(i)}$ is the j -th word of the document. We thus consider the set of all the words from all the documents:

$$\mathcal{X} = (x_1^{(1)}, \dots, x_{c_1}^{(1)}, \dots, x_1^{(N)}, \dots, x_{c_N}^{(N)}).$$

It is easy to see that \tilde{M}_2 can be expressed as a function of \mathcal{X} , for all pairs $u, v \in \{1, \dots, n\}$ we have

$$(\tilde{M}_2)_{u,v}(\mathcal{X}) = \frac{\sum_{i \neq j} (x_i^{(1)})_u (x_j^{(1)})_v + \dots + \sum_{i \neq j} (x_i^{(N)})_u (x_j^{(N)})_v}{C_2}$$

where

$$C_2 = \sum_{i=1}^N c_i (c_i - 1).$$

We now define the following function:

$$\Phi(\mathcal{X}) = \|\tilde{M}_2(\mathcal{X}) - M_2\|_F$$

and observe that, given

$$\mathcal{X} = (x_1^{(1)}, \dots, x_{c_1}^{(1)}, \dots, x_i^{(l)}, \dots, x_1^{(N)}, \dots, x_{c_N}^{(M)})$$

and

$$\mathcal{X}' = (x_1^{(1)}, \dots, x_{c_1}^{(1)}, \dots, x_i^{(l)'}, \dots, x_1^{(N)}, \dots, x_{c_N}^{(M)})$$

we have

$$\begin{aligned} |\Phi(\mathcal{X}) - \Phi(\mathcal{X}')| &\leq \|\tilde{M}_2(\mathcal{X}) - \tilde{M}_2(\mathcal{X}')\|_F = \\ &= \sqrt{\sum_{u,v=1}^n \left(\frac{\sum_{i \neq j} (x_j^{(l)})_u ((x_i^{(l)})_v - (x_i^{(l)'})_v)}{C_2} \right)^2} \leq \frac{\sqrt{2} \max_j(c_j)}{C_2}. \end{aligned}$$

We are now able to apply McDiarmid's inequality stating that

$$\mathbb{P}(\|\tilde{M}_2 - M_2\|_F > \mathbb{E}(\|\tilde{M}_2 - M_2\|_F) + \epsilon) \leq e^{-\frac{\epsilon^2 C_2^2}{(\max_j(c_j))^2 C_1}}$$

So, by setting

$$t = \frac{\epsilon C_2}{\max_j(c_j) \sqrt{C_1}}$$

we get

$$\mathbb{P}(\|\tilde{M}_2 - M_2\|_F > \mathbb{E}(\|\tilde{M}_2 - M_2\|_F) + t \frac{\max_j(c_j) \sqrt{C_1}}{C_2}) \leq e^{-t^2}.$$

We now provide a bound for $\mathbb{E}(\|\tilde{M}_2 - M_2\|_F)$. We begin observing that

$$\tilde{M}_2 = \sum_{i=1}^N \frac{\tilde{M}_2^{(i)}}{N}$$

where $\tilde{M}_2^{(i)}$ are independent matrices defined as follows:

$$(\tilde{M}_2^{(i)})_{(u,v)} = N \frac{\sum_{i \neq j} (x_i^{(i)})_u (x_j^{(i)})_v}{C_2}$$

$$\begin{aligned} \mathbb{E}(\|\tilde{M}_2 - M_2\|_F) &\leq \sqrt{E(\|\tilde{M}_2 - M_2\|_F^2)} \leq \sqrt{E(\|\tilde{M}_2\|_F^2) - \|M_2\|_F^2} = \\ &= \sqrt{\frac{1}{N^2} E(\|\sum_{i=1}^N \tilde{M}_2^{(i)}\|_F^2) - \|M_2\|_F^2} \leq \sqrt{\frac{1}{N^2} E((\sum_{i=1}^N \|\tilde{M}_2^{(i)}\|_F)^2) - \|M_2\|_F^2} = \\ &= \sqrt{\frac{1}{N^2} \sum_{i=1}^N E(\|\tilde{M}_2^{(i)}\|_F^2) - \frac{1}{N} \|M_2\|_F^2} \end{aligned}$$

We now look at $\|\tilde{M}_2^{(i)}\|_F^2$

$$\|\tilde{M}_2^{(i)}\|_F^2 = \frac{N^2}{C_2^2} \sum_{u,v} \left(\sum_{i \neq j} (x_i^{(i)})_u (x_j^{(i)})_v + \sum_{\substack{i \neq j, h \neq l \\ (i,j) \neq (h,l)}} (x_i^{(i)})_u (x_j^{(i)})_v (x_h^{(i)})_u (x_l^{(i)})_v \right)$$

so

$$E(\|\tilde{M}_2^{(i)}\|_F^2) = \frac{N^2}{C_2^2} (c_i(c_i - 1) + c_i(c_i - 1)(c_i(c_i - 1) - 1) \sum_{u,v} \sum_{j=1}^k \mu_{u,j}^2 \mu_{v,j}^2 \omega_k)$$

If we call W the following matrix in $\mathbb{R}^{n \times n}$

$$(W_2)_{u,v} = \sqrt{\sum_{j=1}^k \mu_{u,j}^2 \mu_{v,j}^2 \omega_k}$$

we get

$$\frac{\sum_{i=1}^N E(\|\tilde{M}_2^{(i)}\|_F^2)}{N^2} = \frac{1}{C_2^2} (C_2 + \|W_2\|_F^2 (\sum_{i=1}^N (c_i(c_i - 1))^2 - C_2))$$

obtaining:

$$\mathbb{E}(\|\tilde{M}_2 - M_2\|_F) \leq \sqrt{\frac{1}{C_2} + \|W_2\|_F^2 D_2 - \frac{1}{N} \|M_2\|_F^2}$$

where

$$D_2 = \frac{(\sum_{i=1}^N (c_i(c_i - 1))^2 - C_2)}{C_2^2}$$

we get

$$\mathbb{P}(\|\tilde{M}_2 - M_2\|_F > \sqrt{\frac{1}{C_2} + \|W_2\|_F^2 D_2 - \frac{1}{N} \|M_2\|_F^2} + t \frac{\max_j(c_j) \sqrt{C_1}}{C_2}) \leq e^{-t^2}.$$

In conclusion, we can state that if

$$e^{-t^2} = \delta$$

we get, for any $\delta \in (0, 1]$

$$\mathbb{P}(\|\tilde{M}_2 - M_2\|_F > \epsilon) \leq \delta$$

where

$$\epsilon = \sqrt{\frac{1}{C_2} + \|W_2\|_F^2 D_2 - \frac{1}{N} \|M_2\|_F^2} + \sqrt{\log\left(\frac{1}{\delta}\right) \frac{\max_j(c_j) \sqrt{C_1}}{C_2}}.$$

A similar argument works for M_3 . □

B Proofs for Section 4

B.1 Proof of Lemma 4.1

Proof. We will use the same notation of the proof of Theorem 4.1. We consider any E with rank k satisfying

$$M_2 = EE'$$

and we recall that we can find an isometry O such that

$$EO = M(\text{diag}(\Omega))^{\frac{1}{2}} \quad (13)$$

Now, given a feature r , we can easily define $E_r \in \mathbb{R}^{(n-1) \times k}$ as E with the r -th row removed. It is easy to see that

$$M_{2,r} = E_r E_r'$$

and that there exist an isometry O_r realizing the equation

$$E_r O_r = M_r(\text{diag}(\Omega))^{\frac{1}{2}}.$$

We will prove that, for each r , we have $O = O_r$. First, we define the following notation:

$$\bar{M}_r := M_r(\text{diag}(\Omega))^{\frac{1}{2}}$$

$$\bar{M} := M(\text{diag}(\Omega))^{\frac{1}{2}}.$$

Also, it will be useful to define the rows of E and \bar{M} :

$$E = \begin{pmatrix} e_1 \\ \vdots \\ e_n \end{pmatrix} \in \mathbb{R}^{n \times k}, \quad \bar{M} = \begin{pmatrix} m_1 \\ \vdots \\ m_n \end{pmatrix} \in \mathbb{R}^{n \times k}.$$

We will prove the theorem only for $r = n$, but the proof is very similar for the other cases. By equation (13) we have that

$$\begin{aligned} O &= E^* \bar{M} = (E' E)^{-1} E' \bar{M} = \\ &= (E_r' E_r + e_r' e_r)^{-1} (E_r' \bar{M}_r + e_r' m_r). \end{aligned}$$

Now, using the Sherman–Morrison formula (Sherman and Morrison, 1950) we obtain

$$O = [(E_r' E_r)^{-1} + \frac{(E_r' E_r)^{-1} e_r' e_r (E_r' E_r)^{-1}}{1 + e_r (E_r' E_r)^{-1} e_r'}] (E_r' \bar{M}_r + e_r' m_r)$$

and so

$$\begin{aligned} O &= (E_r' E_r)^{-1} E_r' \bar{M}_r + (E_r' E_r)^{-1} e_r' m_r - \\ &\quad - \frac{(E_r' E_r)^{-1} e_r' e_r (E_r' E_r)^{-1} E_r' \bar{M}_r + (E_r' E_r)^{-1} e_r' e_r (E_r' E_r)^{-1} e_r' m_r}{1 + e_r (E_r' E_r)^{-1} e_r'} = \\ &= O_r + \frac{(E_r' E_r)^{-1} e_r' m_r - (E_r' E_r)^{-1} e_r' e_r (E_r' E_r)^{-1} E_r' \bar{M}_r}{1 + e_r (E_r' E_r)^{-1} e_r'}. \end{aligned}$$

We now just have to prove that

$$m_r = e_r(E'_r E_r)^{-1} E'_r \bar{M}_r = e_r E_r^* \bar{M}_r.$$

We know that

$$M_2 = EE' = \bar{M}\bar{M}'$$

thus

$$\begin{pmatrix} E_r \\ e_r \end{pmatrix} \begin{pmatrix} E'_r & e'_r \end{pmatrix} = \begin{pmatrix} \bar{M}_r \\ m_r \end{pmatrix} \begin{pmatrix} \bar{M}'_r & m'_r \end{pmatrix}$$

so the following system holds:

$$\begin{cases} E_r E'_r = \bar{M}_r \bar{M}'_r \\ e_r E'_r = m_r \bar{M}'_r \end{cases}$$

so

$$E'_r = E_r^* \bar{M}_r \bar{M}'_r$$

from which we have our thesis

$$m_r \bar{M}'_r = e_r E_r^* \bar{M}_r \bar{M}'_r \implies m_r = e_r E_r^* \bar{M}_r.$$

□

C Proofs for Section 5

C.1 Proof of Theorem 5.1

Proof. The goal of the proof is to develop a perturbation bound for each column μ_i of the unknown matrix

$$M = [\mu_1 | \dots | \mu_k]$$

such that, if

$$\tilde{M} = [\tilde{\mu}_1 | \dots | \tilde{\mu}_k]$$

we can state, for a given *Bound* function of ϵ

$$\|\tilde{\mu}_i - \mu_i\|_F^2 \leq \text{Bound}(\epsilon).$$

We notice, from Algorithm 2, at step 8, that each $\tilde{\mu}_i$ is obtained as the diagonal entries of the following matrix:

$$\tilde{O}' \tilde{E}_i^* \tilde{M}_{3,i} (\tilde{E}_i')^* \tilde{O}$$

and so, we will need to find the perturbations of the matrices composing this equation, as the following relation holds:

$$\|\tilde{\mu}_i - \mu_i\|_2 \leq \|\tilde{O}' \tilde{E}_i^* \tilde{M}_{3,i} (\tilde{E}_i')^* \tilde{O} - O' E_i^* M_{3,i} (E_i')^* O\|_F.$$

In short, having perturbation bounds on $\tilde{M}_{3,i}$, \tilde{E}_i^* and \tilde{O} will be sufficient to reach our goal.

Perturbations on $\tilde{M}_{3,i}$

We know by hypotheses that

$$\|\tilde{M}_{3,i} - M_{3,i}\|_F = \|\Delta_{M_{3,i}}\|_F < \epsilon$$

Perturbations on \tilde{E}_i^*

It is a known fact (see Stewart, 1998) that, given the SVD

$$\tilde{M}_2 = \tilde{U}\tilde{S}\tilde{V}, \quad M_2 = USU'$$

if

$$\|\tilde{M}_2 - M_2\|_F < \epsilon,$$

we have that

$$\|\tilde{S} - S\|_F \leq \epsilon. \tag{14}$$

Algorithm 2 considers at step 2 the following approximation of E

$$\tilde{E} = \tilde{U}_k(\tilde{S}_k)^{\frac{1}{2}}$$

while the unperturbed value of E can be found as

$$E = U_k(S_k)^{\frac{1}{2}}$$

where the subscript k indicates the truncation at the k -th singular value. So, to reach a perturbation bound on \tilde{E}_i^* for a given i , we first need to look for a perturbation bound on \tilde{E} , that will be obtained bounding the error of $(\tilde{S}_k)^{\frac{1}{2}}$ and \tilde{U}_k . The first one is a consequence of equation (14):

$$\|\Delta_S\|_F = \|(\tilde{S}_k)^{\frac{1}{2}} - S^{\frac{1}{2}}\|_F < \frac{\epsilon}{2\sqrt{\sigma_k(M_2)}},$$

where $\sigma_k(M_2)$ is the k -th the singular value of M_2 . To find a bound on \tilde{U}_k we will use Lemma C.1 to get:

$$\|\Delta_U\|_F = \|\tilde{U}_k - U_k\|_F \leq 2\frac{\epsilon\|E\|_F^2}{\alpha_M} + O(\epsilon^2).$$

where

$$\alpha_M = \min_{i \neq j} (|\sigma_i(M_2) - \sigma_j(M_2)|)$$

and $\sigma_i(M_2)$ are the singular values of M_2 .

We thus conclude that

$$\tilde{E} = E + \Delta_U S^{\frac{1}{2}} + U \Delta_S + \Delta_U \Delta_S$$

and so

$$\|\tilde{E} - E\|_F < f(\epsilon) = \epsilon(2\frac{\|E\|_F^2\|S^{\frac{1}{2}}\|_F}{\alpha_M} + \frac{\|U\|_F}{2\sqrt{\sigma_k(M_2)}}) + O(\epsilon^2).$$

We are now ready to find a perturbation bound on the pseudoinverse of \tilde{E} after the removal of row i . This can be accomplished using a known bound from (Stewart, 1990): If

$$\|\tilde{E}_i - E_i\|_F < f(\epsilon)$$

then

$$\|\tilde{E}_i^\star - E_i^\star\|_F \leq f(\epsilon)\tau(E)$$

where

$$\tau(E) = \|E^\star\|_F^2 + \|(E'E)^{-1}\|_F \|\mathbb{I} - EE^\star\|_F.$$

Perturbations on \tilde{O}

We now look for the value of $g(\epsilon) = \|O - \tilde{O}\|_F$. In particular, O comes from the decomposition of $E_r^\star M_{3,r}(E_r')^\star$:

$$E_r^\star M_{3,r}(E_r')^\star = O \text{diag}(\mu_r) O'. \quad (15)$$

while \tilde{O} is the set of the left singular vectors of

$$\tilde{E}_r^\star \tilde{M}_{3,r}(\tilde{E}_r')^\star$$

First, we look for a bound on the following value:

$$\|E_r^\star M_{3,r}(E_r')^\star - \tilde{E}_r^\star \tilde{M}_{3,r}(\tilde{E}_r')^\star\|_F.$$

to get

$$\|E_r^\star M_{3,r}(E_r')^\star - \tilde{E}_r^\star \tilde{M}_{3,r}(\tilde{E}_r')^\star\|_F \leq f(\epsilon)\tau(E)\|E^\star\|_F\|M_{3,r}\|_F + \epsilon\|E^\star\|_F^2 + O(\epsilon^2).$$

Similarly to what done in Lemma C.1 below, using (Chen et al., 2012, Remark 2.2) we obtain

$$\begin{aligned} \|\tilde{O} - O\|_F &< g(\epsilon) = \\ &= \frac{2(f(\epsilon)\tau(E)\|E^\star\|_F^3\|M_{3,r}\|_F^2 + \epsilon\|E^\star\|_F^4\|M_{3,r}\|_F)}{\alpha} + O(\epsilon^2). \end{aligned}$$

We are now able to conclude our proof by analyzing

$$\begin{aligned} \|\mu_i - \tilde{\mu}_i\|_2 &\leq \|\tilde{O}' \tilde{E}_i^\star \tilde{M}_{3,i}(\tilde{E}_i')^\star \tilde{O} - O' E_i^\star M_{3,i}(E_i')^\star O\|_F \leq \\ &\leq P_1 g(\epsilon) + P_2 \epsilon + P_3 f(\epsilon) + O(\epsilon^2) \end{aligned}$$

where P_1, P_2 and P_3 are polynomials in $\|E_i\|_F, \|O\|_F$ and $\|M_{3,i}\|_F$. The thesis follows making explicit these polynomials. \square

Lemma C.1. Consider M_2 the perturbed \tilde{M}_2 , and their SVD

$$\tilde{M}_2 = \tilde{U} \tilde{S} \tilde{V}', \quad M_2 = U S U'$$

Let \tilde{U}_k and U_k be matrices of the first k left singular vectors of \tilde{M}_2 and M_2 . If $\|\tilde{M}_2 - M_2\|_F < \epsilon$, then it holds that

$$\|\tilde{U}_k - U_k\|_F < \frac{2\epsilon\|E\|_F^2}{\min_{i \neq j} |\sigma_i(M_2) - \sigma_j(M_2)|} + O(\epsilon^2).$$

where $\sigma_i(M_2)$ are the singular values of M_2 .

Proof. Consider

$$\tilde{M}_2 \tilde{M}_2' = \tilde{U} \tilde{S}^2 \tilde{U}', \quad M_2 M_2' = U S^2 U'.$$

Then

$$\|\tilde{M}_2 \tilde{M}_2' - M_2 M_2'\|_F \leq 2\epsilon\|E\|_F^2 + O(\epsilon^2)$$

Also, \tilde{U} and U are the eigenvectors respectively of $\tilde{M}_2 \tilde{M}_2'$ and $M_2 M_2'$. We are thus able to apply the what says in (Chen et al., 2012, Remark 2.2), to obtain the thesis. \square

References

- Alighieri, D. (1979). *La Divina Commedia, a cura di N. Sapegno*. Nuova Italia, Firenze.
- Anandkumar, A., Ge, R., Hsu, D., Kakade, S. M., and Telgarsky, M. (2014). Tensor decompositions for learning latent variable models. *Journal of Machine Learning Research*, 15(1):2773–2832.
- Anandkumar, A., Hsu, D., and Kakade, S. M. (2012a). A method of moments for mixture models and Hidden Markov models. In *COLT*, volume 1, page 4.
- Anandkumar, A., Liu, Y.-k., Hsu, D. J., Foster, D. P., and Kakade, S. M. (2012b). A spectral algorithm for Latent Dirichlet Allocation. In *Advances in Neural Information Processing Systems*, pages 917–925.
- Balle, B., Hamilton, W. L., and Pineau, J. (2014). Methods of moments for learning stochastic languages: Unified presentation and empirical comparison. In *ICML*, pages 1386–1394.
- Belkin, M. and Sinha, K. (2010). Toward learning gaussian mixtures with arbitrary separation. In *COLT*, pages 407–419. Citeseer.
- Blei, D. M., Ng, A. Y., and Jordan, M. I. (2003). Latent Dirichlet Allocation. *Journal of machine Learning research*, 3(Jan):993–1022.
- Chen, X. S., Li, W., and Xu, W. W. (2012). Perturbation analysis of the eigenvector matrix and singular vector matrices. *Taiwanese Journal of Mathematics*, 16(1):pp–179.
- Dasgupta, S. (1999). Learning mixtures of gaussians. In *Foundations of Computer Science, 1999. 40th Annual Symposium on*, pages 634–644. IEEE.
- Dasgupta, S. and Schulman, L. (2007). A probabilistic analysis of em for mixtures of separated, spherical gaussians. *Journal of Machine Learning Research*, 8(Feb):203–226.
- Dempster, A. P., Laird, N. M., and Rubin, D. B. (1977). Maximum likelihood from incomplete data via the em algorithm. *Journal of the royal statistical society. Series B (methodological)*, pages 1–38.
- Griffiths, T. L. and Steyvers, M. (2004). Finding scientific topics. *Proceedings of the National academy of Sciences*, 101(suppl 1):5228–5235.
- Halko, N., Martinsson, P.-G., and Tropp, J. A. (2011). Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions. *SIAM review*, 53(2):217–288.
- Hsu, D. and Kakade, S. M. (2013). Learning mixtures of spherical gaussians: moment methods and spectral decompositions. In *Proceedings of the 4th conference on Innovations in Theoretical Computer Science*, pages 11–20. ACM.
- Hsu, D., Kakade, S. M., and Zhang, T. (2012). A spectral algorithm for learning Hidden Markov models. *Journal of Computer and System Sciences*, 78(5):1460–1480.
- Jain, P., Jin, C., Kakade, S. M., Netrapalli, P., and Sidford, A. (2016). Matching matrix bernstein with little memory: Near-optimal finite sample guarantees for oja’s algorithm. *arXiv preprint arXiv:1602.06929*.

- Kalai, A. T., Moitra, A., and Valiant, G. (2010). Efficiently learning mixtures of two gaussians. In *Proceedings of the forty-second ACM symposium on Theory of computing*, pages 553–562. ACM.
- Liberty, E. (2013). Simple and deterministic matrix sketching. In *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 581–588. ACM.
- McDiarmid, C. (1989). On the method of bounded differences. *Surveys in combinatorics*, 141(1):148–188.
- Moitra, A. and Valiant, G. (2010). Settling the polynomial learnability of mixtures of gaussians. In *Foundations of Computer Science (FOCS), 2010 51st Annual IEEE Symposium on*, pages 93–102. IEEE.
- Mossel, E. and Roch, S. (2005). Learning nonsingular phylogenies and hidden markov models. In *Proceedings of the thirty-seventh annual ACM symposium on Theory of computing*, pages 366–375. ACM.
- Newman, D., Asuncion, A., Smyth, P., and Welling, M. (2009). Distributed algorithms for topic models. *Journal of Machine Learning Research*, 10(Aug):1801–1828.
- Sanjeev, A. and Kannan, R. (2001). Learning mixtures of arbitrary gaussians. In *Proceedings of the thirty-third annual ACM symposium on Theory of computing*, pages 247–257. ACM.
- Sherman, J. and Morrison, W. J. (1950). Adjustment of an inverse matrix corresponding to a change in one element of a given matrix. *The Annals of Mathematical Statistics*, 21(1):124–127.
- Stewart, G. W. (1990). Matrix perturbation theory.
- Stewart, G. W. (1998). Perturbation theory for the singular value decomposition.
- Van Der Walt, S., Colbert, S. C., and Varoquaux, G. (2011). The numpy array: a structure for efficient numerical computation. *Computing in Science & Engineering*, 13(2):22–30.
- Vempala, S. and Wang, G. (2002). A spectral algorithm for learning mixtures of distributions. In *Foundations of Computer Science, 2002. Proceedings. The 43rd Annual IEEE Symposium on*, pages 113–122. IEEE.
- Zou, J. Y., Hsu, D. J., Parkes, D. C., and Adams, R. P. (2013). Contrastive learning using spectral methods. In *Advances in Neural Information Processing Systems*, pages 2238–2246.